SOFTWARE ASSURANCE FORUM

BUILDING SECURITY IN

Measurement Framework for Software Assurance and Information Security

Nadya Bartol
Bob Martin

October 16, 2008

Homeland Security

Agenda

- SwA Measurement Overview
  - Approaches
  - Implementing Measures
  - Measurement Framework
- What is Measurable
  - CVE, CCE, CWE, and CAPEC
  - CVSS, CCSS, and CWSS
- Measure Development Exercise
- Conclusion

Measurement Overview

Why Measurement

Drivers

- Need to demonstrate the value of SwA
- Decreasing funding and increasing accountability for it
- Calls for quantifiable ROI
- Need for data to support decisions

Benefits

- Supports business case for assurance
- Provides quantifiable information to support decision making and accountability
- Quantifies SwA improvements
- Helps demonstrate regulatory compliance
- Helps demonstrate value to executives
- Motivates stakeholder to change behavior

Response

- Developing *Practical measurement Framework for Software Assurance* that
  - Is harmonized with common system and software and security measurement methodologies
  - Provides an approach for quantifying achievement of SwA goals and objectives within the context of individual projects, programs, or enterprises
  - Provides a framework for the organizations to integrate SwA measurement in their overall measurement efforts in a cost-effective and a seamless manner

Measurement Framework Summary

This document does	This document does not
• Explain how to integrate SwA measurement into existing measurement approaches • Provide a common framework for addressing SwA measurement regardless of what measurement approach is used • Explain a basic process for measurement common to referenced measurement methodologies • Provide example goals/information needs and measures for four primary SwA stakeholder groups • Contain measures based on common enumerations to get to tangible software-related things to measure	• Create a new stand-alone measurement approach for SwA • Provide a single text book for SwA measurement that can be used without referencing other methods • List ALL possible SwA measures that could be ever needed by a project or organization

Base Methodologies

- ISO/IEC 15939, Practical Software and System Measurement (PSM)
- CMMI Measurement and Analysis Process Area
- CMMI Goal, Question, Indicator, Measure (GQIM)
- NIST SP 800-55 Rev1, Performance Measurement Guide for Information Security
- ISO/IEC 27004, Information Security Management Measurement

Measurement Process

Create/ Update Measures	Collect Data	Store Data	Analyze and Compile Data	Report Measures	Use Measures
• State goals • Identify data sources and elements • Analyze how goals and data elements relate • Create a series of measures	• Gather data from available data sources	• Document /store data in an appropriate repository	• Analyze collected data • Compile and aggregate into measures • Interpret data • Identify causes of findings	• Document measures in appropriate reporting formats • Report measures to stakeholders	• Support decisions • Allocate resources • Prioritize improvements • Communicate to executives and external stakeholders

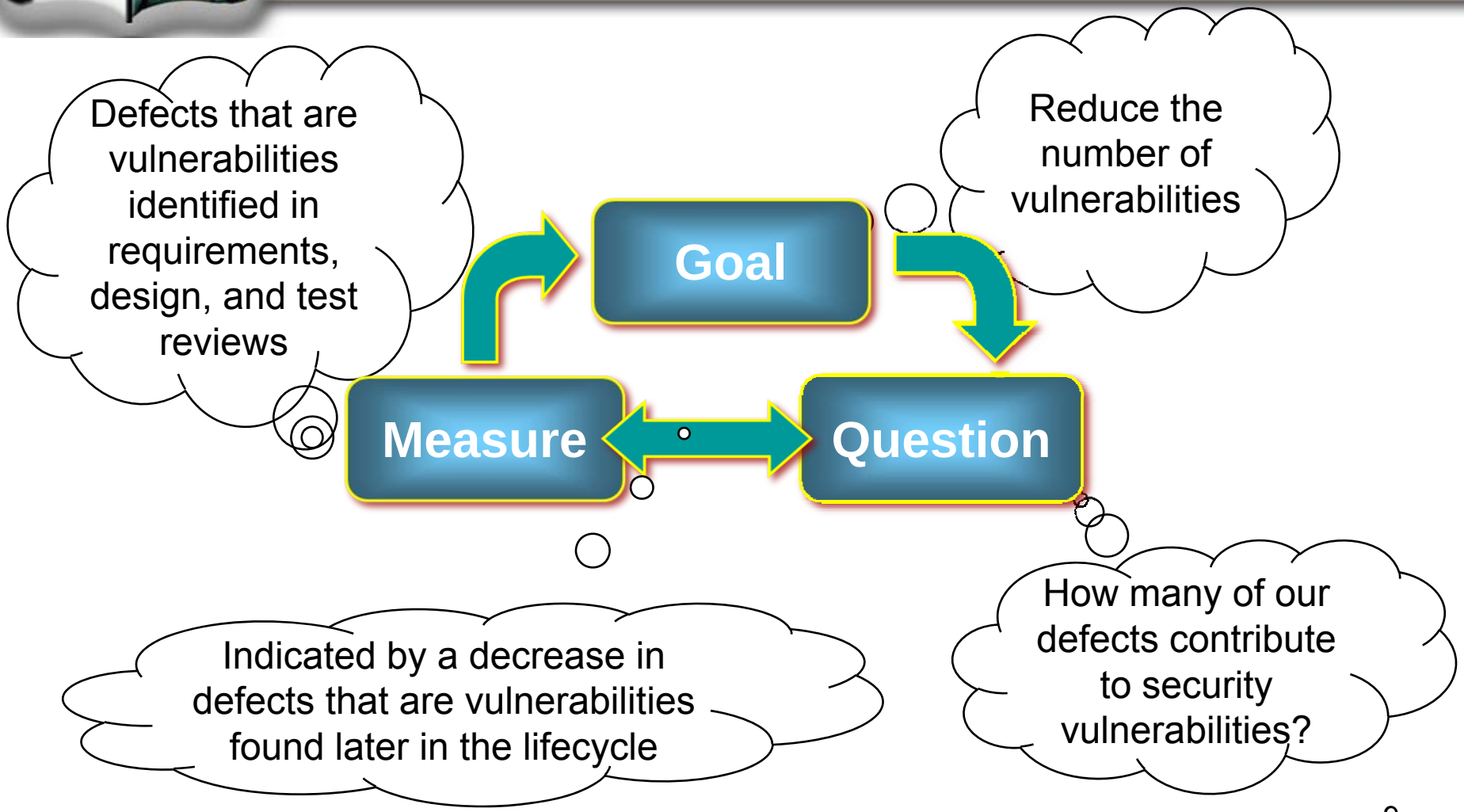
Continuous Improvement

- Refresh measures to ensure they are still relevant to the project, program, or organization
- Train measurement staff

Measurement Framework at a Glance

PSM ISO/IEC 15939	CMMI® (Measurement and Analysis Process Area)	CMMI® GQ(I)M		ISO/IEC 27004	NIST SP 800-55 Revision 1

Developing Measures



What is Measurable

Data Sources for SwA Measurement

Enumerations

- **Common Vulnerabilities and Exposures (CVE)** is a list of identifiers (CVE-IDs) for publicly known vulnerabilities.
- **Common Configuration Enumeration (CCE)**

is a list of identifiers (CCE-IDs) for security related configuration controls for OS platforms and mainstream applications.

- **Common Weakness Enumeration (CWE)**
- **Common Attack Pattern Enumeration and Classification (CAPEC)**

is a enumerated list (CWE-IDs) of the architecture, design, and implementation weaknesses that can lead to exploitable problems in software.

Prioritization "Languages"

- **Common Vulnerability Scoring System (CVSS)**

is a enumerated list (CAPEC-IDs) of the patterns of attacks used by adversaries to go after IT systems.

- **Common Configuration Scoring System (CCSS)**
- **Common Weakness Scoring System (CWSS)**

is a standard structured set of measures to convey vulnerability severity in order to help determine urgency and prioritize response.

will be a standard structured set of measures to convey the severity of software security configuration issues in order to help determine urgency and prioritize response.

will be a standard structured set of measures to convey weakness severity in order to help determine urgency and prioritize response.

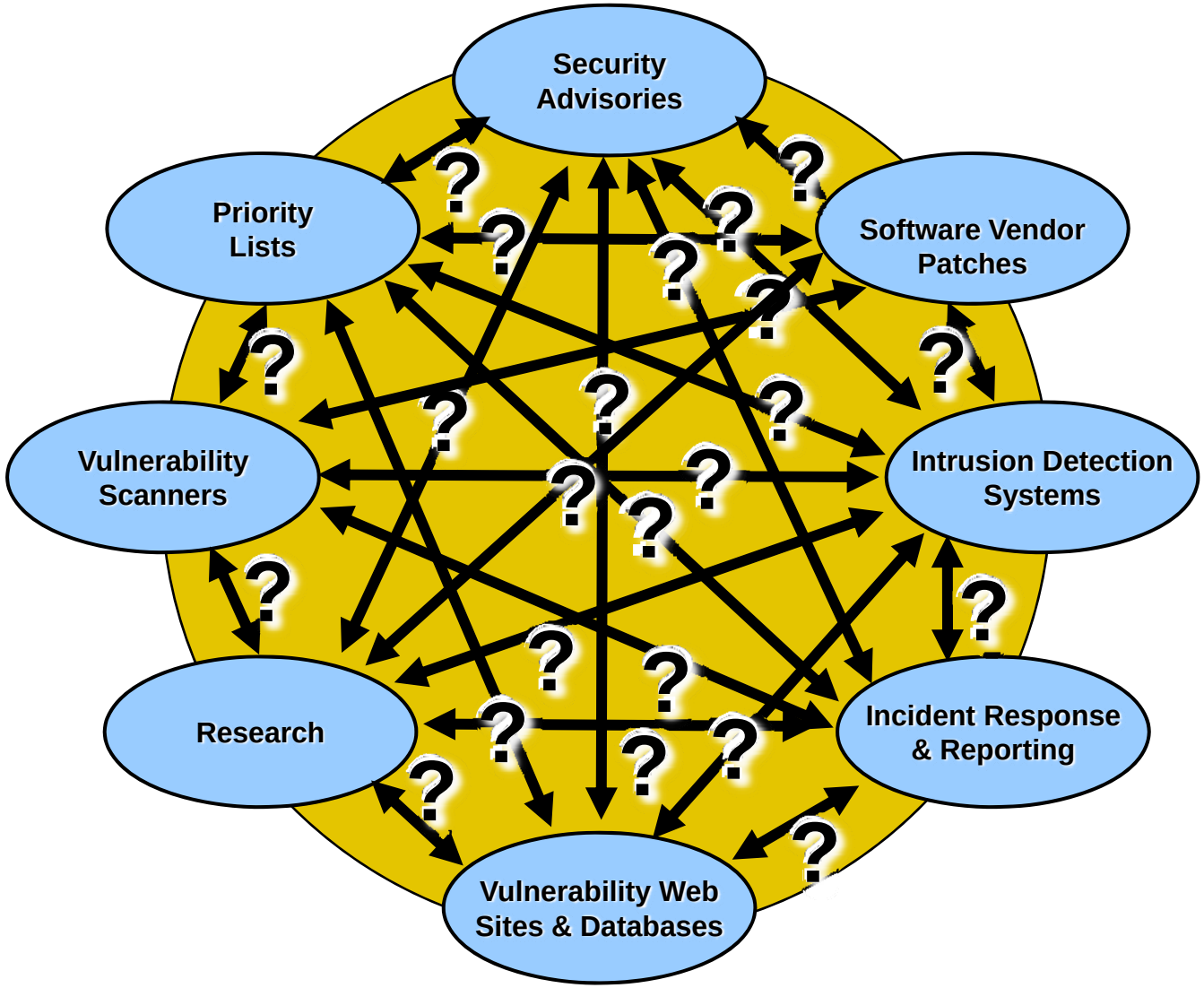
Common Vulnerabilities and Exposures (CVE)

- enables fast, accurate correlation of vulnerability information across the security industry
- Key tenets
  - **one identifier for one vulnerability**
  - **dictionary of standardized descriptions for vulnerabilities and exposures**
  - **publicly accessible for review or download from the Internet**
  - **international scope**
  - **industry participation in open forum (editorial board)**
  - **compatibility program for products & services (~300)**

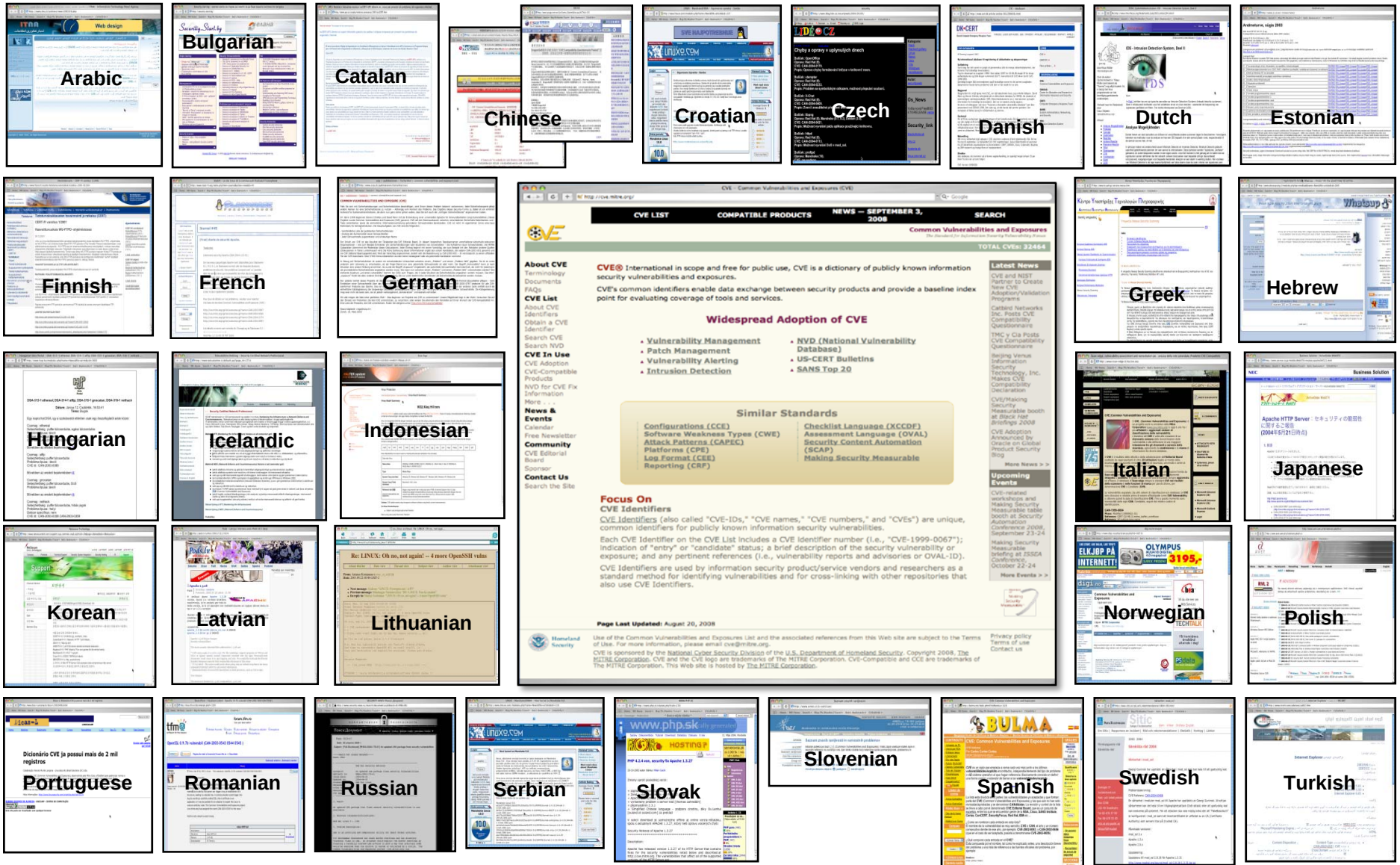
Note: the CVE compatibility program is transitioning to an adoption/validation approach as a NIST/MITRE partnership

Foundation of NIST NVDCollection

Difficult to Integrate Information on Vulnerabilities and Exposures



CVE is Widely Used &Available 32,595 and climbing…



CCE Entries Defined

CCE-W2K-178

Definition: The "restrict guest access to application log" policy should be set correctly.

Technical Mechanism (1 or more):

(1) HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\Application\RestrictGuestAccess
(2) defined by Group Policy

Parameters (1 or more): (1) enabled/disabled

- **Standardized Identifier -Similar to existing CVE and CME**
  - **Final version may have no descriptive info embedded (e.g. W2K)**
- **Definition - Describes the affect on system…**
  - **…But does not assert a recommendation**
- **Technical Mechanisms - Describes mechanisms used to achieve the intended affect**
- **Parameter – Describes range of logical values**

CCE Use in Configuration Guidance

- Written guidance for system configurations
- Primary User: Guide authors
- Actions:
  - Must balance human readability against technical precision
  - Consumers: Administrators, Configuration Management vendors, Configuration Audit vendors

NSA Solaris Guide (XCCDF)
Configure SSH

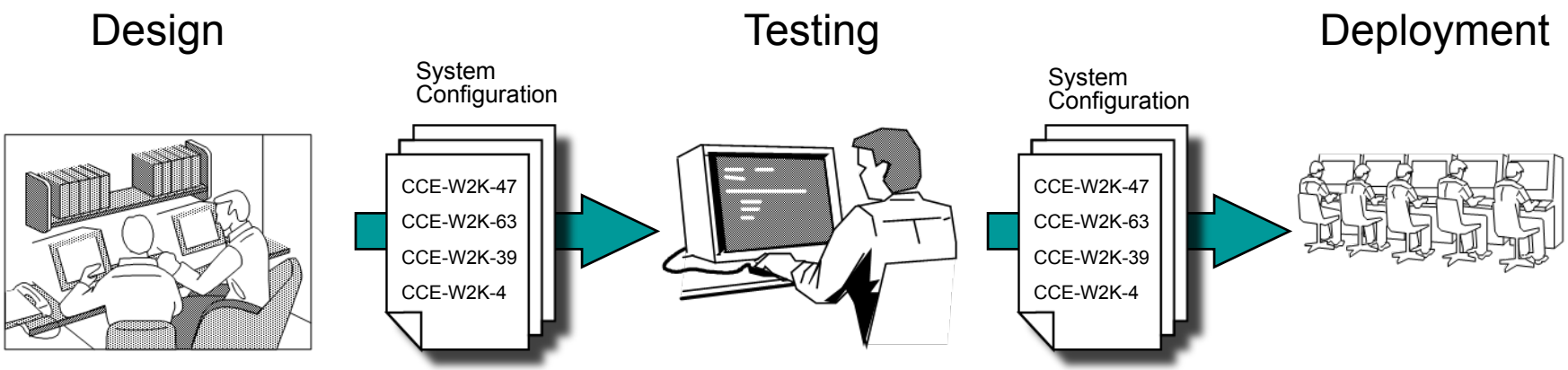
CCE-Sol9-384: SSH protocol 2
OVAL-DEF-ID: 28274
CCE-Sol9-26: SSH rhosts
OVAL-DEF-ID: 18474
CCE-Sol9-178: SSH root login
OVAL-DEF-ID: 29883
CCE-Sol9-179: SSH client configuration
OVAL-DEF-ID: 74736
…

CIS Solaris Bencmark (XCCDF)
Configure SSH

CCE-Sol9-384: SSH uses protocol 2 only?
OVAL-DEF-ID: 78334
CCE-Sol9-178: SSH daemon restricts root login?
OVAL-DEF-ID:99383
CCE-Sol9-179: SSH client has the proper global protocol configuration?
OVAL-DEF-ID:49488
CCE-Sol9-238: SSH daemon maximum authorization tries is properly configure?
OVAL-DEF-ID: 28274
…

CCE Use in Configuration Management

- Proactive approach to vulnerability or system management
- Primary User: IT administration
- Actions:
  - Design: Defined to meet operational & security requirements
  - Testing: Pre-deployment testing of configurations
  - Deployment: Distribution and maintenance of configurations



CCE Use in Configuration Audit

- Testing against “known-goods” (checklists)…
  - not against “known-bads” (vulnerability lists)
- Primary User: Security Audit group
- Actions:
  - Use audit tools to test compliance to external best practices
    - E.g. CIS Benchmarks, NSA Guides, DISA Stigs, NIST Checklists
  - Find checks in audit tools corresponding to requirements

NSA Configuration Issues

…

NSA-0: Accounts not locking out after 3 unsuccessful logon attempts (CCE-W2K-3)

NSA-P: Autorun is enabled (CCE-W2K-205)

NSA-Q: Bad logon count not reset after 15 minutes (CCE-W2K-1)

NSA-R: Last username is displayed in login dialog (CCE-W2K-212)

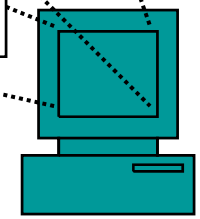
…

COTS Configuration Checks

…

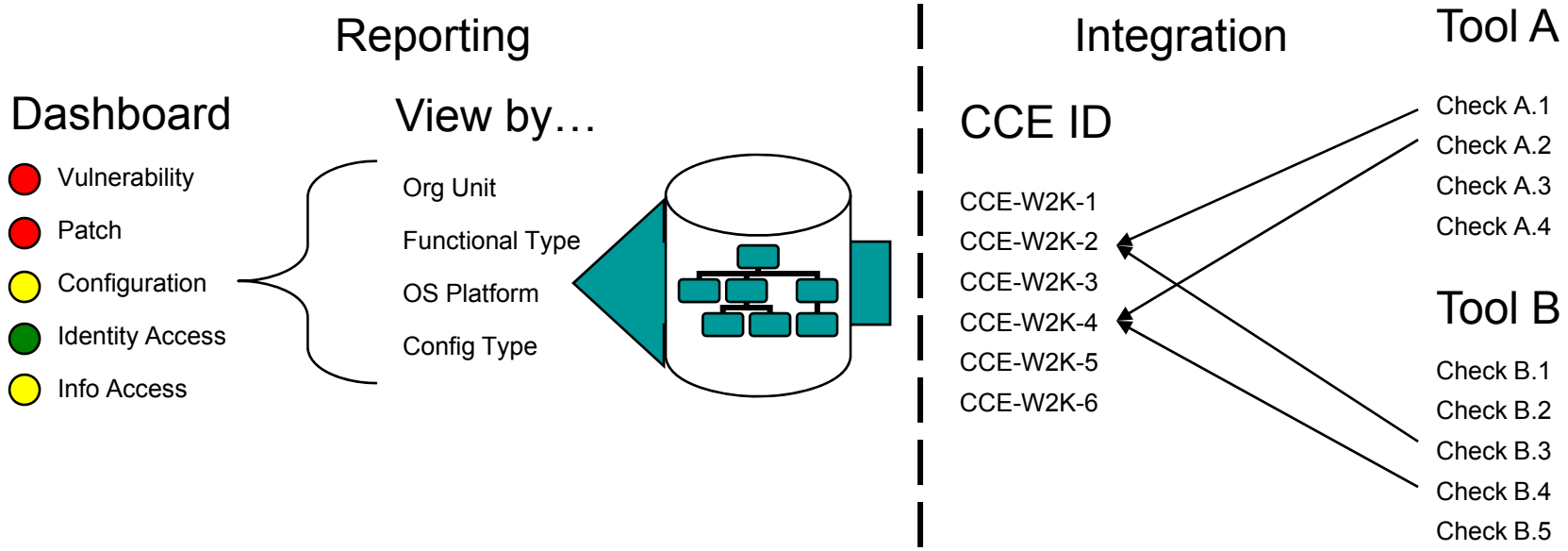
- PolicyLockoutThreshold (CCE-W2K-3)
- RegVal93 (CCE-W2K-205)
- PolicyLockoutObservationWindow (CCE-W2K-1)
- RegVal99; RegVal118 (CCE-W2K-212)

…



CCE Use in Situational Awareness

- Comprehensive view of IT security posture
- Primary User: CISOs, CIOs, Compliance Officers
- Actions:
  - Integration: Combine results from multiple auditing tools
    - E.g. Citadel, DISA Gold Disk, BindView, Symantec, CIS Scoring Tool
  - Reporting: Roll-up and drill down for audit data



CCE Use in Compliance / C&A

- Primary User: CISOs, CIOs, Compliance Officers
- Actions:
  - Demonstrate compliance to high-level requirements
    - E.g. ISO 17799, DoD Directive 8500.2, NIST sp800-53
  - Demonstrate proper utilization of technical best-practices
    - E.g. CIS Benchmarks, DISA Stigs, NSA Guides, Microsoft Guides
  - Must "connect the dots" between requirements and technical controls (configuration settings)
- Example RTM from NIST

CCE provides common language

SECURITY REQUIREMENTS	MAPPING	NEEDED SECURITY CONTROLS
Security Requirement No. 1	1 TO 1	CCE-W2K-35
Security Requirement No. 2	1 TO MANY	CCE-W2K-246, CCE-W2K-99
Security Requirement No. 3 Security Requirement No. 4	MANY TO 1	CCE-W2K-169
Security Requirement No. 5 Security Requirement No. 6	MANY TO MANY	CCE-W2K-243, CCE-W2K-135, CCE-W2K-187

TABLE 2: SAMPLE REQUIREMENTS TRACEABILITY MATRIX

Common Weakness Enumeration (CWE)

CWE Dictionary (1.0)

9/9/08

- dictionary of weaknesses
  - **weaknesses that can lead to exploitable vulnerabilities (i.e. CVEs)**
  - **the things we don't want in our code, design, or architecture**
  - **web site with XML of content, sources of content, and process used**
- structured views
  - **currently provide multiple hierarchical views into all of the CWE dictionary content as well as sub-setting views and slices**
  - **evolving to support dynamic views**
- open community process
  - **to facilitate common terms/ concepts/facts and understanding**
  - **allows for vendors, developers, system owners and acquirers to understand tool capabilities/coverage and priorities**
  - **utilize community expertise**
  - **formal compatibility and effectiveness program (~30)**

Foundation for other DHS, NSA, OSD, NIST, OWASP, SANS, SEI, and OMG SwAEfforts

SOFTWARE ASSURANCE FORUM

BUILDING SECURITY IN

CWE-79 Failure to Sanitize Directives in a Web Page

(aka 'Cross-site scripting' (XSS))

[cwe.mitre.org/data/definition/79.html]

CWE-79: Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (

Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS))

Weakness ID: 79 *(Weakness Base)* Sta

Description

Summary

The software does not sufficiently sanitize user-controllable input for content before it is prepared in output that is used as a

Extended Description

Unsanitized special elements that have control implications in web pages, such as HTML tags or mouse events, are interprete characters that execute in violation of the client's trust in the application or system. This weakness usually enables cross-site attacks in web applications.

Alternate Terms

CSS

"CSS" was once used as the acronym for this problem, but this can cause confusion with "Cascading Style Sheets," so this a declined significantly. Its use is discouraged by CWE.

Likelihood of Exploit

High to Very High

Weakness Ordinalities

Resultant *(where the weakness is typically related to the presence of some other weaknesses)*

Causal Nature

Explicit *(an explicit weakness resulting from behavior of the developer)*

Common Consequences

Confidentiality

The most common attack performed with cross-site scripting involves the disclosure of information stored in user cookies.

Access Control

In some circumstances it may be possible to run arbitrary code on a victim's computer when cross-site scripting is combined flaws.

If successful, cross-site scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be m those of a valid user, compromise confidential information, or execute malicious code on the end user systems for a variety o purposes.

Potential Mitigations

Carefully check each input parameter against a rigorous positive specification (white list) defining the specific characters and fo allowed. All input should be sanitized, not just parameters that the user is supposed to specify, but all data in the request, incl hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is t only fields that are expected to be redisplayed by the site. We often encounter data from the request that is reflected by the ap server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used developer. Therefore, validating ALL parts of the HTTP request is recommended.

This involves "HTML Entity Encoding" all non-alphanumeric characters from data that was received from the user and is now b written to the request.

With Struts, you should write all data from form beans with the bean's filter attribute set to true.

To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support HttpOnly feature (such as Internet Explorer), this attribute prevents the user's session cookie from being accessed by client-si including scripts inserted due to a XSS attack.

Demonstrative Examples

Example 1:

This example covers a Reflected XSS (Type 1) scenario.

Observed Examples

Reference	Description
CVE-2007-5727	Chain: only removes SCRIPT tags, enabling XSS
CVE-2006-4308	Chain: only checks "javascript:" tag

Other Notes

Cross-site scripting weaknesses occur when dynamically generated web pages display input, such as login information, that is not properly validated, allowing an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user that views the site. If successful, cross-site scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on the end user systems for a variety of nefarious purposes.

References

"Cross-site scripting". Wikipedia. 2008-08-26. <http://en.wikipedia.org/wiki/Cross-site_scripting>.

Jeremiah Grossman, Robert "RSnake" Hansen, Petko "pdp" D. Petkov, Anton Rager and Seth Fogie. "XSS Attacks". Syngress. 2007.

M. Howard and D. LeBlanc. "Writing Secure Code". 2nd Edition. Microsoft. 2003.

Relationships

Nature	Type	ID	Name	V	
ChildOf	WG	74	Failure to Sanitize Data into a Different Plane (aka 'Injection')	699 1000	
ChildOf	C	725	OWASP Top Ten 2004 Category A4 - Cross-Site Scripting (XSS) Flaws	711	
ChildOf	C	722	OWASP Top Ten 2004 Category A1 - Unvalidated Input	711	
CanPrecede	WA	494	Download of Untrusted Mobile Code Without Integrity Check	1000	
PeerOf	P	352	Cross-Site Request Forgery (CSRF)	1000	
ChildOf	C	442	Web Problems	699	
ChildOf	WG	20	Insufficient Input Validation	700	
ChildOf	C	712	OWASP Top Ten 2007 Category A1 - Cross Site Scripting (XSS)	629	
MemberOf	V	635	Weaknesses Used by NVD	635	
CanFollow	WA	113	Failure to Sanitize CRLF Sequences in HTTP Headers (aka 'HTTP Response Splitting')	1000	
CanFollow	WA	184	Incomplete Blacklist	1000	692
ParentOf	WA	80	Failure to Sanitize Script-Related HTML Tags in a Web Page (Basic XSS)	699 1000	
ParentOf	WA	81	Failure to Sanitize Directives in an Error Message Web Page	699 1000	
ParentOf	WA	82	Failure to Sanitize Script in Attributes of IMG Tags in a Web Page	699 1000	
ParentOf	WA	83	Failure to Sanitize Script in Attributes in a Web Page	699 1000	
ParentOf	WA	84	Failure to Resolve Encoded URI Schemes in a Web Page	699 1000	
ParentOf	WA	85	Doubled Character XSS Manipulations	699 1000	
ParentOf	WA	86	Failure to Sanitize Invalid Characters in Identifiers in Web Pages	699 1000	
ParentOf	WA	87	Failure to Sanitize Alternate XSS Syntax	699 1000	

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Cross-site scripting (XSS)
7 Pernicious Kingdoms			Cross-site Scripting
CLASP			Cross-site scripting
OWASP Top Ten 2007	A1	Exact	Cross Site Scripting (XSS)
OWASP Top Ten 2004	A1	CWE More Specific	Unvalidated Input
OWASP Top Ten 2004	A4	Exact	Cross-Site Scripting (XSS) Flaws

Applicable Platforms

SOFTWARE ASSURANCE FORUM

BUILDING SECURITY IN

CWE Compatibility & Effectiveness Program

(launched Feb 2007)

cwe.mitre.org/compatible/

CWE – CWE Compatibility

http://cwe.mitre.org/compatible/index.html

AFC Home | MII Home | Search | Map/Ph/Weather/Travel | Bob's Bookmarks | CVEnOVAL | OVAL shared | SPAMmngt | LogoutofSPAMmngt

Common Weakness Enumeration

A community-developed dictionary of common software weaknesses

Home > Compatibility

- CWE List
  - Full Dictionary View
  - Classification Tree
  - Other Views
- About
  - Sources
  - Process
  - Documents
- Community
  - Related Activities
- News
  - Calendar
- Compatibility
  - Program
  - Requirements
  - Make a Declaration
- Contact Us
  - Search the Site

CWE Compatibility

The CWE Compatibility and Effective
reviewed and registered as officially
assisting organizations in their select
assessing their acquired software for
about the various weaknesses and th
education about these issues. Organi
towards compatibility and effectivene

CWE-Compatible Products and Servi
requirements below, while CWE-Effe
requirements. Please review the com
compatibility and effectiveness.

CWE Searchable	— users may
CWE Output	— security el obtain, ass
Mapping Accuracy	— security el identifiers
CWE Documentation	— capability's documentation describes CWE, CWE compatibility, and how CWE-related functionality in the capability is used

Organizations Participating

All organizations participating in the CWE Compatibility and Effectiveness Program are listed below, including those with CWE-Compatible Products and Services and those with Declarations to Be CWE-Compatible.

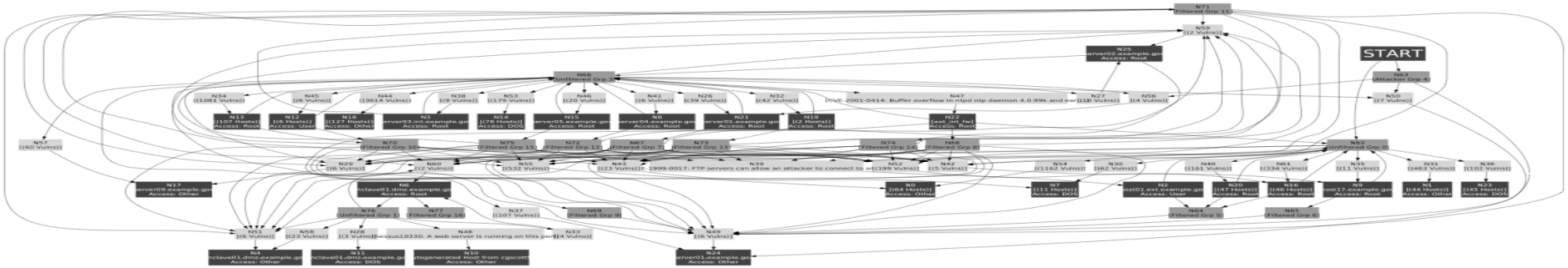
TOTALS
Organizations Participating: **17**
Products & Services: **28**

Products are listed alphabetically by organization name:

armorize technologies — Secure Your Web Code
SECURITY DATABASE — KEEP A VIGILANT EYE ON YOUR DEFENSES
VERACODE
FORTIFY SOFTWARE
CENZIC
SPI DYNAMICS
GRAMMATECH
Klocwork
SANS
SkillBridge
IPA — INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
SofCheck
IBM
WATCHFIRE
checkmarx
cigital
OUNCE LABS
CERIAS — Center for Education and Research in Information Assurance and Security

Attack Patterns Overview

- Represent common approaches to attack
- Abstracted from actual exploits and attacks
- Gives you an attacker's perspective you may not have on your own
- Excellent resource for many key activities
  - Abuse Case development
  - Architecture attack resistance analysis
  - Risk-based security testing
  - Red team penetration testing
- Primarily attack-centric testing methods



What is CAPEC?

- **Community effort targeted at:**
  - Standardizing the capture and description of attack patterns
  - Collecting known attack patterns into an integrated enumeration that can be consistently and effectively leveraged by the community
  - Classifying attack patterns such that users can easily identify the subset of the entire enumeration that is appropriate for their context
- **Future plans:**
  - New patterns
  - Align patterns with other resources
  - Formalize patterns to finer granularity to support test case generation and bridging with the malware and incident response communities

Leveraging attack patterns as test case templates

Attack patterns contain information that can significantly assist in defining contexts, preconditions, test data, action steps, post conditions and variation axes for security test cases

Context:

- Context Description, Examples-Instances, Related Weaknesses, Related Vulnerabilities, Relevant Security Requirements, Relevant Design Patterns, Relevant Security Patterns

Preconditions:

Test Data:

- Attack Prerequisites, Attacker Skill or Knowledge Required, Resources Required

Action Steps:

Post Conditions:

- Description
- Description, Method of Attack, Injection Vector, Payload, Activation Zone
- Description, Attack Motivation-Consequences, Payload Activation Impact

Very simplistic CAPEC test case example

Test Case 1: Single quote SQL injection of registration page web form fields

Test Case Goal: Ensure SQL syntax single quote character entered in registration page web form fields does not cause abnormal SQL behavior

Context:
- This test case is part of a broader SQL injection syntax exploration suite of tests to probe various potential injection points for susceptibility to SQL injection. If this test case fails, it should be followed-up with test cases from the SQL injection experimentation test suite.

Preconditions:
- Access to system registration page exists
- Registration page web form field content are used by system in SQL queries of the system database upon page submission
- User has the ability to enter free-form text into registration page web form fields

Test Data:
- ASCII single quote character

Action Steps:
- Enter single quote character into each web form field on the registration page
- Submit the contents of the registration page

Post Conditions:
- Test case fails if SQL error is thrown
- Test case passes if page submission succeeds without any SQL errors

SQL Injection– capec.mitre.org/data/definitions/66.html

CAPEC - Individual CAPEC Dictionary Definition (Release 1.1)

http://capec.mitre.org/data/definitions/31.html

CAPEC Common Attack Pattern Enumeration and Classificati

A Community Knowledge Resource for Building Secure Software

Home > CAPEC List > Individual CAPEC Dictionary Definition (Release 1.1)

CAPEC List
Full CAPEC Dictionary
Classification Tree
Other Views

About CAPEC
Documents
Resources

Community
Related Activities
Collaboration List

Contact Us
Search the Site

Individual CAPEC Dictionary Definition (Release 1.1)

SQL Injection

Attack Pattern ID 66

Typical Severity High

Description Summary

This attack exploits target software that constructs SQL statements based on user in
constructs SQL statements based on the input, the resulting SQL statement performs actio

SQL Injection results from failure of the application to appropriately validate input. W
without proper validation as part of SQL queries, it is possible to glean information from th
the database and the design of the application, it may also be possible to leverage injectio
choice. SQL Injection enables an attacker to talk directly to the database, thus bypassing t
as well as ability to add or modify data in the database. In order to successfully inject SQL

Attack Execution Flow

Explore

1. **Survey application**: The attacker first takes an inventory of the functionality exposed by the appl

Attack Step Techniques	
Description	
Spider web sites for all available links	
Sniff network communications with application using a utility such as WireShark.	

Outcomes		
ID	Type	Description
c66s0o1	Success	At least one data input to application identified.
c66s0o2	Failure	No inputs to application identified. Note that just because no inputs are ide

Experiment

1. **Determine user-controllable input susceptible to injection**: Determine the user-controllable i is vulnerable to SQL injection, attempt to inject characters that have special meaning in SQL (such etc.). The goal is to create a SQL query with an invalid syntax.

Attack Step Techniques	
Description	
Use web browser to inject input through text fields or through HTTP GET parameters.	
Use a web application debugging tool such as Tamper Data, TamperIE, WebScarab,etc. to modif	
Use network-level packet injection tools such as netcat to inject input	
Use modified client (modified by reverse engineering) to inject input.	

Indicators of Susceptibility			
ID	Type	Description	Environments
c66s1i1	Negative	Attacker receives normal response from server.	env-Web env-ClientServer env-Peer2Peer env-CommProtocol
c66s1i2	Positive	Attacker receives an error message from server indicating that there was a problem with the SQL query.	env-Web env-ClientServer env-Peer2Peer env-CommProtocol
c66s1i3	Negative	Server sends a specific error message that indicates programmatic parsing of the input data (e.g. NumberFormatException)	env-Web env-ClientServer env-Peer2Peer env-CommProtocol

Outcomes		
ID	Type	Description
c66s1o1	Success	At least one user-controllable input susceptible to injection found.
c66s1o2	Failure	No user-controllable input susceptible to injection found.

Security Controls		
ID	Type	Description
c66s1sc1	Detective	Search for and alert on unexpected SQL keywords in application logs (e.g. SELECT, DROP, etc.).
c66s1sc2	Preventative	Input validation of user-controlled data before including it in a SQL query
c66s1sc3	Preventative	Use parameterized queries (e.g. PreparedStatement in Java, and Command.Parameters.Add() to set query parameters in .NET)

2. **Experiment and try to exploit SQL Injection vulnerability**: After determining that a given input is vulnerable to SQL Injection, hypothesize what the underlying query looks like. Iteratively try to add logic to the query to extract information from the database, or to modify or delete information in the database.

Attack Step Techniques	
Description	Environments
Use public resources such as "SQL Injection Cheat Sheet" at http://ferruh.mavituna.com/makale/sql-injection-cheatsheet/, and try different approaches for adding logic to SQL queries.	env-Web env-ClientServer env-Peer2Peer env-CommProtocol
Add logic to query, and use detailed error messages from the server to debug the query. For example, if adding a single quote to a query causes an error message, try ; "' OR 1=1; --", or something else that would syntactically complete a hypothesized query. Iteratively refine the query.	env-Web env-ClientServer env-Peer2Peer env-CommProtocol
Use "Blind SQL Injection" techniques to extract information about the database schema.	env-Web env-ClientServer env-Peer2Peer env-CommProtocol
If a denial of service attack is the goal, try stacking queries. This does not work on all platforms (most notably, it does not work on Oracle or MySQL). Examples of inputs to try include: "'; DROP TABLE SYSOBJECTS; --" and "'); DROP TABLE SYSOBJECTS; --". These particular queries will likely not work because the SYSOBJECTS table is generally protected.	env-Web env-ClientServer env-Peer2Peer env-CommProtocol

Related Weaknesses

CWE-ID	Weakness Name	Weakness Relationship Type
89	Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')	Targeted
74	Failure to Sanitize Data into a Different Plane (aka 'Injection')	Secondary
20	Insufficient Input Validation	Secondary
390	Detection of Error Condition Without Action	Secondary

Related Attack Patterns

ID	Name	Relationship Type	Relationship Description
7	Blind SQL Injection	More Abstract	

Relevant Security Requirements
Special characters in user-controllable input must be escaped before use by the application.
Only use parameterized stored procedures to query the database.
Input data must be revalidated in the parameterized stored procedures.
Custom error pages must be used to handle exceptions such that they do not reveal any information about the architecture of the application or the database.

Related Security Principles
- Reluctance to Trust
- Failing Securely
- Defense in Depth

Related Guidelines
- Never Use Input as Part of a Directive to any Internal Component
- Handle All Errors Safely

Purpose
Penetration
Exploitation

CIA Impact

Confidentiality Impact	Integrity Impact	Availability Impact
High	High	High

Technical Context

Architectural Paradigm	Framework	Platform	Language
All	All	All	All

References
CWE - SQL Injection
CWE - Input Validation
CWE - Improper Error Handling

Source

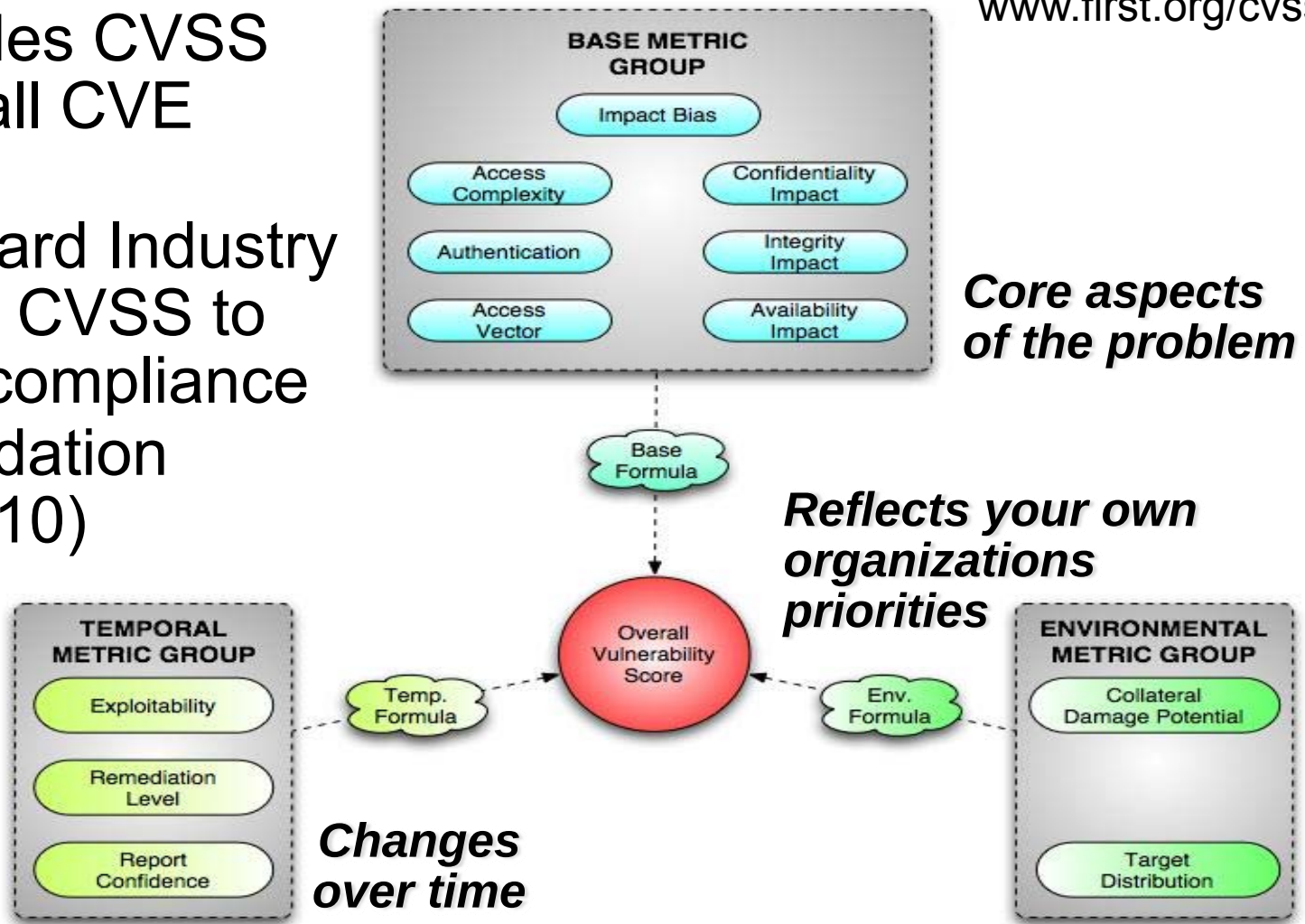
Submission(s)			
Submitter	Organization	Date	Comment
Chiradeep B Chhaya		2007-01-12	Second Draft

Modification(s)			
Modifier	Organization	Date	Comment
Malik Hamro	Cigital, Inc	2007-02-27	Reformat to new schema and review
Sean Barnum	Cigital, Inc	2007-03-05	Review and revise
Richard Struse	VOXEM, Inc	2007-03-26	Review and feedback leading to changes in Description and Related Attack Patterns
Sean Barnum	Cigital, Inc	2007-04-13	Modified pattern content according to review and feedback

Common Vulnerability Scoring System

www.first.org/cvss/

- NVD provides CVSS scores for all CVE identifiers
- Payment Card Industry (PCI) using CVSS to determine compliance
- Formal validation program (~10)



Core aspects of the problem

Reflects your own organizations priorities

Changes over time

Common Configuration Scoring System

- Based on the same metric groupings as CVSS…base, temporal, and environment

Common Configuration Scoring System (CCSS)

Karen Scarfone, NIST

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

NIST Interagency Report 7502 (Draft)

The Common Configuration Scoring System (CCSS) (DRAFT)

Karen Scarfone
Peter Mell

CCSS

Current State of CCSS

- Draft specification for base metrics and formulas
- Not started on temporal or environmental metrics
- Initial assumption that temporal metrics may not be applicable to CCSS
  - From CVSS: availability of exploit code, availability of remediation, confidence in vulnerability reports
- Environmental metrics work to be done in conjunction with review of CVSS metrics

csrc.nist.gov/publications/drafts/nistir-7502/Draft-NISTIR-7502.pdf

Common Configuration Scoring System

4.1 CCE-4675-5

Consider CCE-4675-5 for Sun Solaris 10: "Kernel level auditing should be enabled or disabled as appropriate." If auditing should be enabled but is not, a wide range of events will not be logged, including login/logout events, administrative actions, file attribute modifications, and process events.

Since some of the type
Vector is "Network".
automatically fail to be
Authentication metric
integrity. There is no i

The base vector for thi
of 5.0.

4.5 CCE-2366-3

Consider CCE-2366-3 for Windows XP: "The 'shut down the system' user right should be assigned to the correct accounts." We do not know to whom the access has been granted.

The obvious case is that users have permission to shut down the host, but should not. Access to this file requires a local account on the computer, so the Access Vector is "Local". The Access Complexity is "Low" because the user can simply use regular OS features to shut down the host. No additional authentication other than the initial target access is required to use the weakness, so the Authentication metric is "None". Giving this privilege to any users is a full compromise of host availability (the host can be shut down by users at will). There is no impact on confidentiality or integrity. The base vector for this weakness is: AV:L/AC:L/Au:N/C:N/I:N/A:C. This vector produces a base score of 4.9.

Another case with possible security implications is if no users, including administrators, had the right to shut the host down. This could delay administrators in shutting down the host when needed, so it would be a partial compromise of host availability. The base vector would be AV:L/AC:L/Au:N/C:N/I:N/A:P. This vector produces a base score of 2.1.

Common Weakness Scoring System

cwe.mitre.org/cwss/

- Will use a metric / component approach similar to CVSS but…
- The temporal component will take into account things like level of public awareness of the weakness, level of public knowledge about the weakness, availability of tools that scan for the weakness, availability of tools that attack the weakness, level of observed attacks on the weakness in the wild, etc.
    - For example, buffer overflows and SQL injection might get higher scores because they're currently very well-known, whereas "SQL column truncation" is brand-new and thus less likely to be exploited by a general attacker.
- The environment component will need to reflect the application to application variation due to what the application's function is and how accessible its attack surface is and any questions about exploitability.
- The base component of CWSS, unlike CVSS's third-party analysis of public vulnerabilities a la NVD, can use a more granular approach that reflects the fact that each developer knows their code inside and out and can make use of a more detailed scoring calculation for the base score.

Measure Development Exercise

What We Will Do

- Leverage the Framework to develop several measures that
  - Are included in the document
  - Use enumerations
  - Can be adopted in today's environment

Items To Specify

- Goal/Information Need
- Entities
- Attributes
- Measures
- Formulas
- Frequency
- Responsible persons
- Visualization
- Interpretation

Example 1

Measures	Information Need	Benefit
• Number of relevant attack patterns covered by executed test cases • Density of test cases identified and executed per relevant attack pattern	• Ascertain that testing is conducted against all relevant attack patterns	• To ensure that testing has been conducted against all attacks relevant to the system, including all relevant steps, techniques, and varieties

Example 2

Measures	Information Need	Benefit
• Number and percent of applicable defects (weaknesses - known CWEs and vulnerabilities – known CVEs) remediated before the system is operational of total universe of applicable defects that could have been introduced throughout development	• Understanding of SwA that the system provides	• Understand the level of risk and potential liability generated by acquired/integrated product

Example 3

Measures	Information Need	Benefit
• Number and percent of known vulnerabilities (CVEs) discovered post-implementation that could have been remediated before implementation by impact of exploitation • Number and percent of relevant high impact vulnerabilities (CVEs) present in the system • Percent exploitable CVEs that were mitigated through various types of mitigating strategies, such as patches and service packs and mitigating controls	• Identify vulnerabilities exploitation of which would have an unacceptable impact on organization's mission • Ascertain that all appropriate mitigating strategies have been collectively applied	• Better ability to prioritize resources for fixing vulnerabilities • Focus vulnerability mitigation to exploitable vulnerabilities vs. all vulnerabilities regardless of their applicability • Provides insight into cost and impact of SDLC implementation on business and mission

Measure 1

Goal/Information Need	
Entity	
Attributes	
Base Measures	
Formula	
Derived Measures	
Formula	
Indicator	
Decision Criteria	
Interpretation	

Measure 2

Goal/Information Need	
Entity	
Attributes	
Base Measures	
Formula	
Derived Measures	
Formula	
Indicator	
Decision Criteria	
Interpretation	

Measure 3

Goal/Information Need	
Entity	
Attributes	
Base Measures	
Formula	
Derived Measures	
Formula	
Indicator	
Decision Criteria	
Interpretation	

Conclusion

What Did We Learn

- Was this easy to do?
- What was difficult?
- Why was it difficult?
- What can you take from here and use?

What Can You Do?

- Take a look at the resources available at
  - Developing Practical Framework for Software Assurance and Information Security Measurement (https://buildsecurityin.us-cert.gov/swa/downloads/SwA_Measurement.pdf)
  - Populating a web site of software assurance measurement resources (https://buildsecurityin.us-cert.gov/swa/measact.html)
  - Making Security Measurable (http://makingsecuritymeasurable.mitre.org) to learn how to utilize the standards-based information in commercially available security measurement & management tools.
- Become a part of the collaborative solutions, share lessons learned and jointly work obstacles
  - Pilot the measures
  - Come back and tell us about your experience

Contact Info

- Nadya Bartol, CISSP, ISSPCS
  Co-Chair DHS SwA Measurement Working Group
  bartol_nadya@bah.com

- Bob Martin
  ramartin@mitre.org
  http://measurablesecurity.mitre.org/